Minimum Average Distance Triangulations

László Kozma

Universität des Saarlandes, Saarbrücken, Germany
kozma@cs.uni-saarland.de

Abstract. We study the problem of finding a triangulation T of a planar point set S such as to minimize the expected distance between two points x and y chosen uniformly at random from S. By distance we mean the length of the shortest path between x and y along edges of T, with edge weights given as part of the problem. In a different variant of the problem, the points are vertices of a simple polygon and we look for a triangulation of the interior of the polygon that is optimal in the same sense. We prove that a general formulation of the problem in which the weights are arbitrary positive numbers is strongly NP-complete. For the case when all weights are equal we give polynomial-time algorithms. In the end we mention several open problems.

1 Introduction

The problem addressed in this paper is a variant of the classical *network design* problem. In many applications, the average routing cost between pairs of nodes is a sensible network characteristic, one that we seek to minimize. If costs are *additive* (e.g., time delay) and *symmetric*, an edge-weighted, undirected graph G is a suitable model of the connections between endpoints. The task is then to find a spanning subgraph T of G that minimizes the average distance. Johnson *et al.* [1] study the problem when the total edge weight of T is required to be less than a given budget constraint. They prove this problem to be NP-complete, even in the special case when all weights are equal and the budget constraint forces the solution to be a *spanning tree*.

Here we study the problem in a planar embedding: vertices of G are points in the plane, edges of G are straight segments between the points and weights are given as part of the problem. Instead of limiting the total edge weight of the solution, we require the edges of T to be non-intersecting. From a theoretical point of view this turns out to be an essential difference: the problem now has a geometric structure that we can make use of. As an application we could imagine that we wanted to connect n cities with an optimal railroad network using straight line connections and no intersections. We now give a more precise definition of the problem.

Given a set of points $S = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, and weights $w : S^2 \to \mathbb{R}$, having $w(x, x) = 0$ and $w(x, y) = w(y, x)$, for all $x, y \in S$, we want to find a *geometric*, *crossing-free* graph T with vertex set S and edge weights given by w, such that the expected distance between two points chosen uniformly at random from S

is as small as possible. By *distance* we mean the length of the shortest path in T and we denote it by d_T. Since adding an edge cannot increase a distance, it suffices to consider *maximal* crossing-free graphs, i.e., *triangulations*. We call this the MINIMUM AVERAGE DISTANCE TRIANGULATION (MADT) problem.

The previous formulation, if we omit the normalizing factor, amounts to finding a triangulation T that minimizes the following quantity:

$$\mathcal{W}(T) = \sum_{1 \le i < j \le n} d_T(p_i, p_j).$$

Similarly, we ask for the triangulation T of the interior of a polygon with n vertices that has the minimum value $\mathcal{W}(T)$. In this case the triangulation consists of all boundary edges and a subset of the diagonals of the polygon.

We note that in mathematical chemistry, the quantity $\mathcal{W}(T)$ is a widely used characteristic of molecular structures, known as Wiener index [2,3]. Efficient computation of the Wiener index for special graphs, as well as its combinatorial properties have been the subject of significant research [4,5,6].

Optimal triangulations. Finding optimal triangulations with respect to various criteria has been intensively researched in the past decades [7,8]. One particularly well-studied problem is *minimum weight triangulation* (MWT). For polygons, the solution of MWT is found by the $\mathcal{O}(n^3)$ algorithm due to Gilbert [9] and Klincsek [10], a classical example of dynamic programming. For point sets and Euclidean weights MWT was proven to be NP-hard by Mulzer and Rote [11]. For unit weights MWT is trivial since all triangulations have the same cost.

In contrast to both MWT and budgeted network design [1], MADT is interesting even for unit weights. In case of simple polygons, the problem is neither trivial, nor NP-hard. The algorithm we give in § 2.2 uses dynamic programming but it is much more involved than the $\mathcal{O}(n^3)$ algorithm for MWT. Surprisingly, the ideas of the MWT algorithm do not seem to directly carry over to the MADT problem, which, in fact, remains open for Euclidean weights, even for polygons. What makes our criterion of optimality somewhat atypical is that it is highly nonlocal. It is nontrivial to decompose the problem into smaller parts and known techniques do not seem to help.

Our results. We study triangulations of point sets and of polygons. In the case of equal weights on all allowed edges, we assume w.l.o.g. that the weights are equal to one and we refer to the distance as *link distance*. Using link distance, the solution is easily obtained when one point or one vertex can be connected to all the others. This is shown in § 2.1. For the more general case of simple polygons (when no vertex can be connected to all other vertices) in § 2.2 we give an algorithm with a running time of $\mathcal{O}(n^{11})$ that uses dynamic programming. Our approach exploits the geometric structure of the problem, making a decomposition possible in this case.

For general point sets and arbitrary positive, symmetric weights (not necessarily obeying the triangle inequality), in § 2.3 we prove the problem to be

strongly NP-complete, ruling out the existence of an efficient exact algorithm or of a fully polynomial time approximation scheme (FPTAS), unless P=NP. The hardness proof is a gadget-based reduction from PLANAR3SAT. Again, the non-locality of the cost function makes the reduction somewhat difficult, requiring a careful balancing between the magnitudes of edge weights and the size of the construction.

We leave the problem open in the case of Euclidean weights but we present the results of computer experiments for certain special cases in § 3.

2 Results

2.1 Link distance with one-point-visibility

We call a point set S *one-point-visible* if one of the points $p \in S$ can be connected to all the points $q \in S$, where $p \neq q$, using straight segments that do not contain a point of S, except at endpoints. This condition is less restrictive than the usual *generality* (no three points collinear). Similarly, we call a polygon *one-vertex-visible* if one of the vertices can be connected to all others with diagonals or boundary edges of the polygon. The set of *one-vertex-visible* polygons includes all convex polygons. A *fan* is a triangulation in which one point or vertex (called the *fan handle*) is connected to all other points or vertices.

Theorem 1. *For a one-vertex-visible polygon every fan triangulation has the same average distance and this is the smallest possible. For a one-point-visible point set every fan triangulation has the same average distance and this is the smallest possible.*

Proof. The smallest possible distance of 1 is achieved for exactly $2n-3$ pairs of vertices in polygons and $3n-h-3$ pairs of points in point sets (with h points on the convex hull), for every triangulation (these are the pairs that are connected with an edge and all triangulations of the same polygon or point set have the same number of edges). In a fan triangulation, all remaining pairs are at distance 2 from each other: the path between two vertices not connected with an edge can go via the fan handle. □

2.2 Link distance in simple polygons

We now look at polygons that do not admit a fan triangulation. It would be desirable to decompose the problem and deal with the parts separately. The difficulty lies in the fact that when we are triangulating a smaller piece of the polygon, the decisions affect not just the distances within that piece but also the distances between external vertices. We need to do some bookkeeping of these global distances, but first we make some geometric observations.

Assume that an optimum triangulation T has been found. We use a clockwise ordering of the vertices p_1 to p_n and we denote by p_d be the third vertex of the triangle that includes p_1p_n. Let us visit the vertices from p_1 to p_d in clockwise

order (if p_1p_d is a boundary edge, then we have no other vertices in between). Let p_a be the last vertex in this order such that $d_T(p_a, p_1) < d_T(p_a, p_d)$. There has to be such a vertex, since p_1 itself has this property and p_d does not. Let p_c be the first vertex for which $d_T(p_c, p_d) < d_T(p_c, p_1)$. Again, such a vertex clearly exists (in a degenerate case we can have $p_1 = p_a$ or $p_c = p_d$ or both). Let p_b denote the vertex (other than p_n) that is connected to both p_1 and p_d (unless p_1p_d is on the boundary).

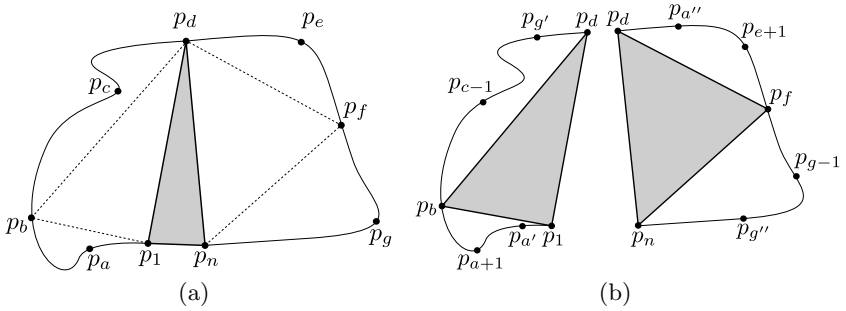

Fig. 1: (a) Special vertices of the polygon. (b) Splitting up the polygon.

On the other side of the triangle $p_1p_dp_n$ we similarly visit the vertices from p_d to p_n and assign the label p_e to the last vertex such that $d_T(p_e, p_d) < d_T(p_e, p_n)$ and p_g to the first vertex such that $d_T(p_g, p_n) < d_T(p_g, p_d)$ and we let p_f be the vertex connected to both p_d and p_n (Fig. 1(a)). Now we can observe some properties of these vertices.

Lemma 1. *Let $1 \leq k \leq d$. Then the following hold (analogous statements hold for $d \leq k \leq n$):*

(a) $d_T(p_k, p_1) < d_T(p_k, p_d)$ *iff* $1 \leq k \leq a$.
(b) $d_T(p_k, p_1) > d_T(p_k, p_d)$ *iff* $c \leq k \leq d$.
(c) $d_T(p_k, p_1) = d_T(p_k, p_d)$ *iff* $a < k < c$. *In particular, if p_b exists, then $a < b < c$. Otherwise $a = 1$, $c = d = 2$, and p_1p_2 is on the boundary.*

Proof. (a) The largest index k for which $d_T(p_k, p_1) < d_T(p_k, p_d)$ is $k = a$ by the definition of p_a. For the converse, observe that for all intermediary vertices p_k on a shortest path between p_1 and p_a we have $d_T(p_k, p_1) < d_T(p_k, p_d)$. Now suppose there is a vertex p_l, with $1 \leq l \leq a$, such that $d_T(p_l, p_d) \leq d_T(p_l, p_1)$. Such an inequality also holds for all intermediary vertices on the shortest path between p_l and p_d. Since the shortest path between p_l and p_d intersects the shortest path between p_a and p_1, the common vertex has to be at the same time strictly closer to p_1 and closer or equal to p_d, a contradiction.

(b) Similar argument as for *(a)*.

(c) First, observe that $a < c$, otherwise some vertex would have to be strictly closer to both p_1 and p_d, a contradiction. Then, since for $1 \leq k < c$ we have

$d_T(p_k, p_1) \leq d_T(p_k, p_d)$ and for $a < k \leq d$ we have $d_T(p_k, p_d) \leq d_T(p_k, p_1)$, it follows that for indices in the intersection of the two intervals ($a < k < c$) we have $d_T(p_k, p_d) = d_T(p_k, p_1)$. The converse follows from (a) and (b). Also, we have $d_T(p_b, p_1) = d_T(p_b, p_d) = 1$. □

Equipped with these facts, we can split the distance between two vertices on different sides of the $p_1p_dp_n$ triangle into locally computable components.

Let $1 \leq x \leq d$. Consider the shortest path between p_x and p_d. Clearly, for all vertices p_k on this path $1 \leq k \leq d$ holds, otherwise the path would go via the edge p_1p_n and it could be shortened via p_1p_d. Similarly, given $d \leq y \leq n$, for all vertices p_k on the shortest path between p_y and p_n, we have $d \leq k \leq n$. We conclude that $d_T(p_x, p_d)$ and $d_T(p_y, p_n)$ only depend on the triangulations of $(p_1, \ldots, p_d)$ and $(p_d, \ldots, p_n)$ respectively. We now express the global distance $d_T(p_x, p_y)$ in terms of these two local distances.

Lemma 2. *Let $p_1, \ldots, p_n$ defined as before, $1 \leq x \leq d$, and $d \leq y \leq n$, and let $\phi = d_T(p_x, p_d) + d_T(p_y, p_n)$. Then the following holds, covering all possible values of x and y:*

$$d_T(p_x, p_y) = \begin{cases} \phi - 1 & \text{if } d \leq y \leq e; \\ \phi + 1 & \text{if } g \leq y \leq n \text{ and } a < x \leq d; \\ \phi & \text{otherwise .} \end{cases}$$

Proof. In each of the cases we use Lemma 1 to argue about the possible ways in which the shortest path can cross the triangle $p_1p_dp_n$. For example, if $d \leq y \leq e$, the shortest path goes through p_d, therefore we have $d_T(p_x, p_y) = d_T(p_x, p_d) + d_T(p_y, p_d)$. Since $d_T(p_y, p_d) = d_T(p_y, p_n) - 1$, we obtain $d_T(p_x, p_y) = \phi - 1$. The other cases use similar reasoning and we omit them for brevity. □

Lemma 2 allows us to decompose the problem into parts that can be solved separately. We proceed as follows: we *guess* a triangle $p_1p_dp_n$ that is part of the optimal triangulation and we use it to split the polygon in two. We also *guess* the special vertices p_a, p_c, p_e, p_g. We recursively find the optimal triangulation of the two smaller polygons with vertices $(p_1, \ldots, p_d)$ and $(p_d, \ldots, p_n)$. Besides the distances *within* the subpolygons we also need to consider the distances *between* the two parts. Using Lemma 2 we can decompose these distances into a distance to p_d in the left part, a distance to p_n in the right part and a constant term.

We now formulate an extended cost function $\mathcal{W}_{\text{EXT}}$, that has a second term for accumulating the distances to endpoints that result from splitting up global distances. The coefficient $\alpha \in \mathbb{N}$ will be uniquely determined by the sizes of the polygons, which in turn are determined by the choice of the index d. We express this new cost function for a general subpolygon $(p_i, \ldots, p_j)$:

$$\mathcal{W}_{\text{EXT}}(T, \alpha)\Big|_i^j = \sum_{i \leq x < y \leq j} d_T(p_x, p_y) + \alpha \sum_{i \leq x \leq j} d_T(p_x, p_j).$$

Observe that minimizing $\mathcal{W}_{\text{EXT}}(T, 0)\Big|_1^n$ solves the initial problem. Using Lemma 2 we can split the sums and the distances until we can express $\mathcal{W}_{\text{EXT}}$ recursively

in terms of smaller polygons and the indices of special vertices a, c, e, g. Note that p_a, p_c, p_e, p_g play the same role as in the earlier discussion, but now the endpoints are p_i and p_j instead of p_1 and p_n.

$$\begin{aligned}\mathcal{W}_{\mathrm{EXT}}(T,\alpha)\Big|_i^j &= \sum_{i\le x<y\le d} d_T(p_x,p_y) + \sum_{d\le x<y\le j} d_T(p_x,p_y) + \sum_{\substack{i\le x\le d\\ d\le y\le j}} d_T(p_x,p_y)\\ &- \sum_{i\le x\le j} d_T(p_x,p_d) + \alpha \sum_{i\le x\le d} d_T(p_x,p_j) + \alpha \sum_{d\le x\le j} d_T(p_x,p_j) - \alpha\cdot d_T(p_d,p_j)\\ &= \mathcal{W}_{\mathrm{EXT}}(T,\alpha+j-d)\Big|_i^d + \mathcal{W}_{\mathrm{EXT}}(T,\alpha+d-i)\Big|_d^j\\ &\quad + (\alpha+j-g+1)(d-a-1) + (e-d+1)(i-d).\end{aligned}$$

How can we make sure that the constraints imposed by the choice of the special vertices p_a, p_c, p_e, p_g are respected by the recursive subcalls? If the left side of the triangle is on the boundary ($d = i+1$), it follows that $a = i$ and $c = d$ and it is trivially true that $d_T(p_a,p_i) < d_T(p_a,p_d)$. Similarly, if $d = j-1$, it follows that $e = d$ and $g = j$, therefore $d_T(p_e,p_d) < d_T(p_e,p_j)$. The general case remains, when one of the sides of the triangle is not on the boundary. The following lemma establishes a necessary and sufficient condition for the constraints to hold. We write it only for the side p_ip_d and special indices a and c, a symmetric argument works for the side p_dp_j and special indices e and g.

Lemma 3. *Let $p_i,\dots,p_j$ be a triangulated polygon. Assume that the triangulation contains the triangles $p_ip_dp_j$ and $p_ip_bp_d$. Then the following hold:*

(a) We have a as the largest index ($i \le a \le d$) for which $d_T(p_a,p_i) < d_T(p_a,p_d)$ iff $a+1$ is the smallest index ($i \le a+1 \le b$) such that $d_T(p_{a+1},p_b) < d_T(p_{a+1},p_i)$.

(b) We have c as the smallest index ($i \le c \le d$) for which $d_T(p_c,p_d) < d_T(p_c,p_i)$ iff $c-1$ is the largest index ($b \le c-1 \le d$) such that $d_T(p_{c-1},p_b) < d_T(p_{c-1},p_d)$.

Proof. (a) If a is the largest index such that $d_T(p_a,p_i) < d_T(p_a,p_d)$ then $a+1$ is the smallest index such that $d_T(p_{a+1},p_d) \le d_T(p_{a+1},p_i)$. Since $a+1 \le b$, the shortest path between p_{a+1} and p_d contains p_b, therefore $d_T(p_{a+1},p_b) < d_T(p_{a+1},p_i)$. To see that $a+1$ is the smallest index with this property, we need to prove that $d_T(p_k,p_b) \ge d_T(p_k,p_i)$ for all $i \le k \le a$. This inequality follows from $d_T(p_k,p_d) > d_T(p_k,p_i)$ and $d_T(p_k,p_d) = d_T(p_k,p_b)+1$.

For the converse, assume $a+1$ to be the smallest index such that $d_T(p_{a+1},p_b) < d_T(p_{a+1},p_i)$. Then $d_T(p_a,p_b) \ge d_T(p_a,p_i)$. Since $d_T(p_a,p_d) = d_T(p_a,p_b)+1$, it follows that $d_T(p_a,p_i) < d_T(p_a,p_d)$. To see that a is the largest index with this property, we need $d_T(p_k,p_i) \ge d_T(p_k,p_d)$ for all $a < k \le b$ (for $k > b$ the inequality clearly holds). Again, this follows from $d_T(p_k,p_i) > d_T(p_k,p_b)$ and $d_T(p_k,p_d) = d_T(p_k,p_b)+1$.

(b) Similarly. □

Procedure EXT (see Fig. 2) returns the cost $\mathcal{W}_{\text{EXT}}$ of a triangulation that minimizes this cost. We can modify our procedure without changing the asymptotic running time, such as to return the actual triangulation achieving minimum cost. The results are then merged to form the full solution.

Lemma 3 tells us that in all recursive calls two of the four special vertices are fixed and we only need to guess the remaining two. We label these new vertices as p'_a, p'_g and p''_a, p''_g. They play the same role in their respective small polygons as a and g in the large polygon (see Fig. 1(b) for illustration). The notation $p \leftrightarrow q$ indicates the condition that vertices p and q see each other within the polygon.

---

procedure EXT $\big((p_i, \ldots, p_j),\ p_a, p_c, p_e, p_g,\ \alpha\big)$:
  if $(a = i)$ and $(c = e = i + 1)$ and $(g = j = i + 2)$:
    return $3 + 2\alpha$; /* *the polygon has only three vertices* */
  else:

$$\textbf{return} \quad \min_{\substack{p_d, p'_a, p'_g, p''_a, p''_g: \\ i \le a' \le a+1 \le c-1 \le g' \le d \\ d \le a'' \le e+1 \le g-1 \le g'' \le j \\ p_i \leftrightarrow p_d \leftrightarrow p_j}} \Big\{ \text{EXT}\big((p_i, \ldots, p_d),\ p'_a, p_{a+1}, p_{c-1}, p'_g,\ \alpha + j - d\big) + \text{EXT}\big((p_d, \ldots, p_j),\ p''_a, p_{e+1}, p_{g-1}, p''_g,\ \alpha + d - i\big) + (\alpha + j - g + 1)(d - a - 1) + (e - d + 1)(i - d) \Big\};$$

Fig. 2: Procedure for finding the triangulation that minimizes $\mathcal{W}_{\text{EXT}}(T, \alpha)$.

The process terminates, since every subcall is on polygons of smaller sizes and we exit the recursion on triangles. Clearly every triangulation can be found by the algorithm and the correctness of the decomposition is assured by Lemma 2. The fact that indices a, c, e, g indeed fulfill their necessary role (and thus the expressions in the cost-decomposition are correct) is guaranteed by Lemma 3.

For the cases when there is no suitable vertex that can be connected to the required endpoints and whose index fulfills the required inequalities, we adopt the convention that the minimum of the empty set is $+\infty$, thus abandoning those branches in the search tree.

Theorem 2. *The running time of* EXT *on a polygon with n vertices is $\mathcal{O}(n^{11})$.*

Proof. Observe that all polygons in the function calls have contiguous indices, therefore we can encode them with two integers between 1 and n. Furthermore, if the initial call has $\alpha = 0$, then it can be shown that on each recursive call the parameter α becomes "n minus the number of vertices in the polygon". For this reason it is superfluous to pass α as an argument. There are four remaining parameters which can all take n different values. We can build up a table containing the return values of all $\mathcal{O}(n^6)$ possible function calls, starting from the smallest and ending with the full polygon. In computing one entry we take the minimum of $\mathcal{O}(n^5)$ values, giving a total running time of $\mathcal{O}(n^{11})$. □

2.3 Arbitrary positive weights

We now prove that the decision version of MADT for point sets is NP-complete if the edge weigths w are arbitrary positive numbers, such that $w(p_i, p_j) = 0$ iff $i = j$ and $w(p_i, p_j) = w(p_j, p_i)$ for all i, j. Such a weight function is called a *semimetric*. We leave open the status of the problem for *metric* weights (that obey the triangle inequality). We defer most of the details of the proofs in this section to the Appendix. Where possible, we give a short, intuitive explanation.

MADT (decision version): *For given $\mathcal{W}^\star \in \mathbb{R}$, is there a triangulation T of a given point set S and weights w, such that $\mathcal{W}(T) \leq \mathcal{W}^\star$?*

The problem is clearly in NP, since for a given triangulation T, we can use an all-pairs shortest path algorithm to compute $\mathcal{W}(T)$ and compare it with $\mathcal{W}^\star$ in polynomial time.

We prove NP-hardness using a reduction from PLANAR3SAT [12]. In 3SAT, given a 3-CNF formula, we ask whether there exists an assignment of truth values to the variables such that each clause has at least one *true* literal. PLANAR3SAT restricts the question to planar formulae: those that can be represented as a planar graph in which vertices represent both variables and clauses of the formula and there is an edge between clause C and variable x iff C contains either x or $\neg x$.

Knuth and Ragunathan [13] observed that PLANAR3SAT remains NP-complete if it is restricted to formulae embedded in the following fashion: variables are arranged on a horizontal line with three-legged clauses on the two sides of the line. Clauses and their three legs are properly nested, i.e., none of the legs cross each other. We can clearly have an actual embedding in which all three legs of a clause are straight lines and the "middle" leg is perpendicular to the line of the variables (Appendix: Fig. 7). For simplicity, let us call such an embedding of a formula a *planar circuit*.

We put two extra conditions on the admissible planar circuits such that PLANAR3SAT remains NP-complete when restricted to planar circuits obeying these conditions: (**R1**) no variable appears more than once in the same clause, and (**R2**) every variable appears in at least two clauses.

Lemma 4. *Given a planar circuit ϕ_1, we can transform it into a planar circuit ϕ_2 that obeys* R1 *and* R2*, such that ϕ_2 has a satisfying assignment iff ϕ_1 does.*

Proof. We examine every possible way in which R1 or R2 can be violated and give transformations that remove the violations while preserving the planar embedding, as well as the satisfiability of the circuit. □

The gadgets used in the reduction are shown in Fig. 3. They consist of points in the plane and the weights of the potential edges between them. Weights can take one of three values: the value 1, a small value ε and a large value σ (higher than any distance in the triangulation). We call edges with weight σ *irrelevant* and we do not show them in the figures. Including irrelevant edges

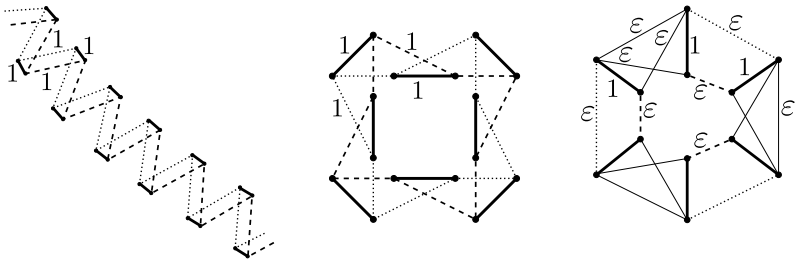

Fig. 3: (a) Wire gadget. (b) Simplified variable gadget. (c) Clause gadget.

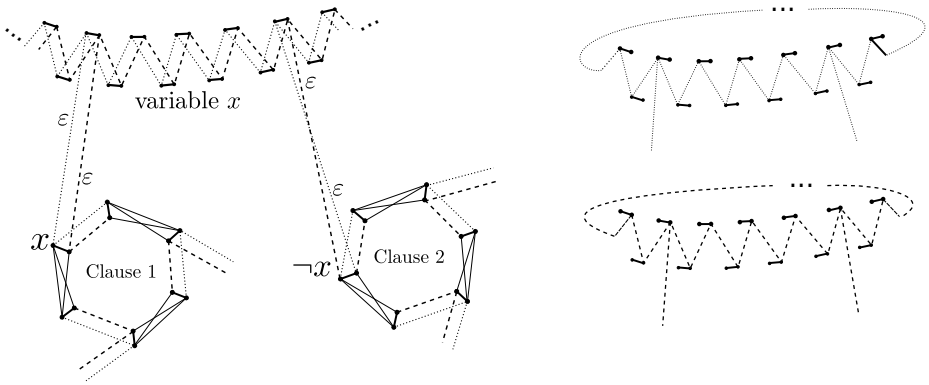

Fig. 4: (a) Bridge between variable and clause. (b) Pure triangulations of a variable.

in a triangulation never decreases the cost $\mathcal{W}$, therefore we can safely ignore them. The values ε and σ depend on the problem size (number of clauses and variables).

The basic building block is the *wire*, shown in Fig. 3(a). The thick solid edges are part of all triangulations. From each pair of intersecting dotted and dashed edges exactly one is included in any triangulation. The weights of all non-irrelevant edges in a wire are 1. The wire-piece can be bent and stretched freely as long as we do not introduce new crossings or remove a crossing between two edges (irrelevant edges do not matter).

The *variable* is a wire bent into a loop, with the corresponding ends glued together. We illustrate this in Fig. 3(b) using a wire-piece with 16 vertices. The construction works with any $4k$ vertices for $k \geq 3$ and in fact we will use *much* more than 16 vertices in each variable.

The *clause* gadget and the weights of its edges are shown in Fig. 3(c).

A *bridge* is a pair of edges that links a variable to a clause. A clause gadget has three fixed "places" where bridges are connected to it. We use *parallel* or *crossing* bridges, as seen in Fig. 4(a). Given a Planar3SAT instance (a planar circuit), we transform it into an instance of MADT as follows: we replace the vertices of the planar circuit by variable- or clause gadgets and we replace the

edge between clause C and variable x by a parallel bridge if C contains x and by a crossing bridge if C contains $\neg x$.

Lemma 5. *Using the gadgets and transformations described above we can represent any planar circuit as a* MADT *instance.* □

Since the gadgets allow a large amount of flexibility, the proof is straightforward. Now we can formulate our main theorem:

Theorem 3. *We can transform any planar circuit ϕ into a* MADT *instance consisting of a point set S in the plane, a* semimetric *weight function $w : S^2 \to \mathbb{R}$ and a threshold $\mathcal{W}^\star$, such that S admits a triangulation T with $\mathcal{W}(T) \leq \mathcal{W}^\star$ iff ϕ has a satisfying assignment. All computations can be done in polynomial time and the construction is of polynomial size, as are all parameters.*

Corollary 1. MADT *with semimetric weights is strongly NP-complete.*

The proof of Theorem 3 relies on a sequence of lemmas. The high level idea is the following: we call a triangulation of the construction *pure*, if every variable gadget, together with its associated bridges contains either only dashed edges or only dotted edges (besides the thick solid edges), see Fig. 4(b). First we show that we only need to consider *pure* triangulations and thus we can use the pure states of the gadgets to encode an assignment of truth values, with the convention *dotted* $\to$ $(true)$, and *dashed* $\to$ $(false)$. Then, we prove that satisfying assignments lead to triangulations with the smallest cost. Finally, we bound the difference in cost between different satisfying assignments and we show how to generate a *baseline* triangulation, with cost not far from the cost of a satisfying assignment (if one exists). The cost of the baseline triangulation can be computed in polynomial time.

We denote the number of variables in our planar circuit by n_v and the number of clauses by n_c. Due to condition R2, we have $n_v \leq 1.5n_c$. We denote the number of vertices in a variable gadget between two bridges (not including the bridge endpoints) by N (the same value for all variables). In Fig. 4(a), for instance, we have $N = 14$. The proof requires a careful balancing of the parameter N describing the size of the construction and the weight ε.

Lemma 6. *If $N > 5 \cdot 10^5 {n_c}^3$, for any impure triangulation T_{impure} of the construction we can find a pure triangulation T_{pure} such that $\mathcal{W}(T_{pure}) < \mathcal{W}(T_{impure})$.*

Proof. The main idea is that if a variable is impure then the loop of the gadget is necessarily broken in some place and this leads to a penalty in cost that cannot be offset by any other change in the triangulation. □

Lemma 7. *Let T_{SAT} be a triangulation corresponding to a satisfying assignment of a planar circuit (assuming such an assignment exists) and let T_{nonSAT} be the triangulation with smallest cost $\mathcal{W}(T_{nonSAT})$ among all triangulations corresponding to nonsatisfying assignments. Then, for $N > 5 \cdot 10^5 {n_c}^3$ and $\varepsilon < \frac{1}{N^2}$, we have $\mathcal{W}(T_{nonSAT}) - \mathcal{W}(T_{SAT}) \geq \frac{N^2}{32}$.* □

Lemma 8. *If T_{SAT1} and T_{SAT2} are two triangulations corresponding to different satisfying assignments, then, given previous bounds on N and ε, we have that* $|\mathcal{W}(T_{SAT1}) - \mathcal{W}(T_{SAT2})| \leq 150{n_c}^2 N$. □

Lemma 9. *For any $T_{baseline}$ and any T_{SAT}, given previous bounds, we have* $|\mathcal{W}(T_{baseline}) - \mathcal{W}(T_{SAT})| \leq 150{n_c}^2 N$. □

We can now generate the threshold as $\mathcal{W}^\star = \mathcal{W}(T_{\text{baseline}}) + 300{n_c}^2 N + 1$. In accordance with our previous constraints we set $N = 10^6{n_c}^3$, $\varepsilon = \frac{1}{10^{13}{n_c}^6}$, and this ensures that $W^\star$ falls in the gap between satisfying and nonsatisfying triangulations. We note that all constructions and computations can be performed in polynomial time and all parameters have polynomial magnitude. This concludes the proof of NP-completeness. □

3 Open questions

The main unanswered question is the status of the problem for metric, in particular Euclidean weights. For polygons we have not succeeded in establishing results similar to Lemma 1 that would enable a dynamic programming approach. For point sets we suspect that the problem remains NP-hard, but we have not found a reduction.

The Euclidean-distance problem remains open even for the special case of regular polygons with n vertices (in this case the boundary of the polygon already gives a $\frac{\pi}{2}$-approximation). Computer simulation shows that the exact solution is the fan for n up to 18, *except* for the cases $n = 7$ and $n = 9$, where the solutions are shown in Fig. 5(a). We conjecture these to be the only counterexamples. As another special case, Fig. 5(b) shows the solutions obtained for a 2-by-n grid, for n up to 16.

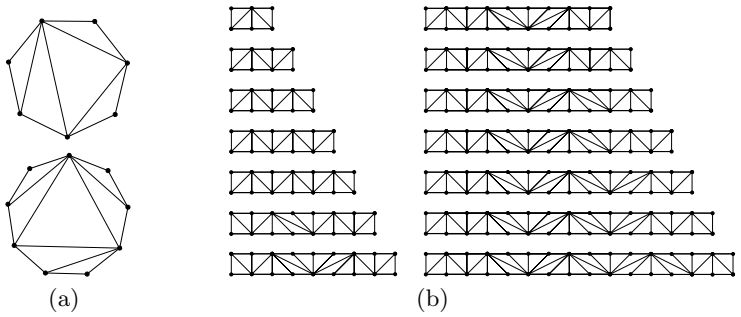

Fig. 5: (a) Solution for regular polygons. (b) Solution for 2-by-n grids.

The problem remains open with unit weights in the case of point sets not admitting a fan, even in special cases such as a 3-by-n grid. For the NP-hard variant

of the problem the question remains whether a polynomial-time approximation scheme (PTAS) exists (an FPTAS is impossible, unless P=NP).

Variants of the problem that we have not studied include placing various constraints on the allowed geometric graphs *besides* non-crossing, such as a total budget on the sum of edge weights or bounded vertex-degree, as well as the case when Steiner points are allowed. One can also study the problem of maximizing the average distance, instead of minimizing it.

4 Acknowledgement

I thank my advisor, Raimund Seidel, for mentioning the problem at the INRIA-McGill-Victoria Workshop on Computational Geometry (2011) at the Bellairs Research Institute, and for valuable comments. I also thank several anonymous reviewers for pointing out errors and suggesting improvements.

References

1. Johnson, D.S., Lenstra, J.K., Kan, A.H.G.R.: The complexity of the network design problem. Networks **8**(4) (1978) 279–285
2. Wiener, H.: Structural Determination of Paraffin Boiling Points. J. of the Am. Chem. Soc. **69**(1) (1947) 17–20
3. Rouvray, D.H.: Predicting chemistry from topology. Sci. Am. **255** (1986)
4. Mohar, B., Pisanski, T.: How to compute the Wiener index of a graph. J. of Mathematical Chemistry **2** (1988) 267–277
5. Dobrynin, A.A., Entringer, R., Gutman, I.: Wiener index of trees: Theory and applications. Acta Applicandae Mathematicae **66** (2001) 211–249
6. Nilsen, C.W.: Wiener index and diameter of a planar graph in subquadratic time (2009)
7. Aurenhammer, F., Xu, Y.: Optimal triangulations. In: Encyclopedia of Optimization. (2009) 2757–2764
8. Bern, M.W., Eppstein, D.: Mesh generation and optimal triangulation. In: Computing in Euclidean Geometry. (1992)
9. Gilbert, P.D.: New results in planar triangulations. Report R-850, Coordinated Sci. Lab., Univ. Illinois, Urbana, IL (1979)
10. Klincsek, G.T.: Minimal triangulations of polygonal domains. Discr. Math. **9** (1980) 121–123
11. Mulzer, W., Rote, G.: Minimum-weight triangulation is NP-hard. J. ACM **55** (2008) 11:1–11:29
12. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2) (1982) 329–343
13. Knuth, D.E., Raghunathan, A.: The problem of compatible representatives. SIAM J. Discr. Math. **5** (1992) 422–427

5 Appendix

Illustration of Theorem 1.

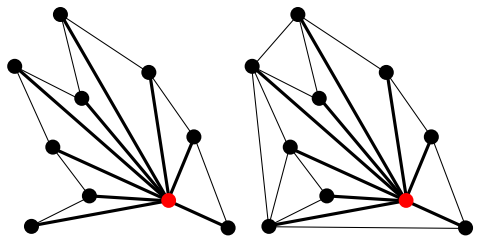

Fig. 6: (a) Fan triangulation of a polygon. (b) Fan triangulation of a point set.

Details of the splitting up of the cost-function using Lemma 2:

$$
\begin{aligned}
\mathcal{W}_{\mathrm{EXT}}(T,\alpha)\Big|_i^j &= \sum_{i\le x<y\le j} d_T(p_x,p_y) + \alpha \sum_{i\le x\le j} d_T(p_x,p_j) \\
&= \sum_{i\le x<y\le d} d_T(p_x,p_y) + \sum_{d\le x<y\le j} d_T(p_x,p_y) - \sum_{d\le x\le j} d_T(p_x,p_d) + \sum_{\substack{i\le x\le d\\ d\le y\le j}} d_T(p_x,p_y) \\
&\quad - \sum_{i\le x\le d} d_T(p_x,p_d) + \alpha \sum_{i\le x\le d} d_T(p_x,p_j) + \alpha \sum_{d\le x\le j} d_T(p_x,p_j) - \alpha\cdot d_T(p_d,p_j) \\
&= \mathcal{W}_{\mathrm{EXT}}(T,0)\Big|_i^d + \mathcal{W}_{\mathrm{EXT}}(T,0)\Big|_d^j - \left(\sum_{d\le x\le j} d_T(p_x,p_j) - (e-d+1) + (j-g+1)\right) \\
&\quad + \left(\sum_{\substack{i\le x\le d\\ d\le y\le j}} \big(d_T(p_x,p_d) + d_T(p_y,p_j)\big) - (d-i+1)(e-d+1) + (j-g+1)(d-a)\right) \\
&\quad - \sum_{i\le x\le d} d_T(p_x,p_d) + \alpha\left(\sum_{i\le x\le d} d_T(p_x,p_d) + (d-a)\right) + \alpha \sum_{d\le x\le j} d_T(p_x,p_j) - \alpha \\
&= \mathcal{W}_{\mathrm{EXT}}(T,\alpha+j-d)\Big|_i^d + \mathcal{W}_{\mathrm{EXT}}(T,\alpha+d-i)\Big|_d^j \\
&\quad + (\alpha+j-g+1)(d-a-1) + (e-d+1)(i-d).
\end{aligned}
$$

Proof. (Lemma 4)

First we give the transformations that establish conditions R1 and R2 while preserving the satisfiability of the formula. We use the following notation: x and y are variables of the original formula, a, b, c and d are variables introduced during the transformation. A clause $(x \vee y \vee z)$ is denoted simply as (xyz) and $\emptyset$ means no clause. The values *True* and *False* are denoted by T and F, respectively.

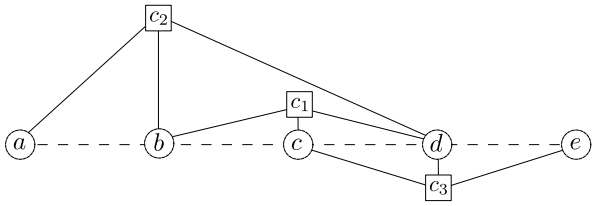

Fig. 7: Planar circuit of the formula $(b \vee \neg c \vee \neg d) \wedge (a \vee \neg b \vee d) \wedge (\neg c \vee d \vee e)$.

The following replacements cover all possible scenarios in which a variable is repeated:

$$(xxx) \to (xab) \wedge (x\neg ab) \wedge (x\neg bc) \wedge (x\neg b\neg c) \qquad (x = T)$$
$$(xx\neg x) \to \emptyset \qquad (x \text{ arbitrary})$$
$$(x\neg x\neg x) \to \emptyset \qquad (x \text{ arbitrary})$$
$$(\neg x\neg x\neg x) \to (\neg xab) \wedge (\neg x\neg ab) \wedge (\neg x\neg bc) \wedge (\neg x\neg b\neg c) \qquad (x = F)$$
$$(xxy) \to (xy\neg a) \wedge (abc) \wedge (a\neg bc) \wedge (a\neg cd) \wedge (a\neg c\neg d) \qquad ((x = T) \text{ or } (y = T))$$
$$(\neg x\neg xy) \to (\neg xy\neg a) \wedge (abc) \wedge (a\neg bc) \wedge (a\neg cd) \wedge (a\neg c\neg d) \qquad ((x = F) \text{ or } (y = T))$$
$$(xx\neg y) \to (x\neg y\neg a) \wedge (abc) \wedge (a\neg bc) \wedge (a\neg cd) \wedge (a\neg c\neg d) \qquad ((x = T) \text{ or } (y = F))$$
$$(\neg x\neg x\neg y) \to (\neg x\neg y\neg a) \wedge (abc) \wedge (a\neg bc) \wedge (a\neg cd) \wedge (a\neg c\neg d) \qquad ((x = F) \text{ or } (y = F))$$
$$(x\neg xy) \to \emptyset \qquad (x, y \text{ arbitrary})$$
$$(x\neg x\neg y) \to \emptyset \qquad (x, y \text{ arbitrary})$$

If x appears in a single clause, we add two new clauses:

$$\emptyset \to (xab) \wedge (xab) \qquad (x \text{ arbitrary})$$

Now we show that the transformations maintain the nesting of the clauses, i.e., they do not introduce crossings. Removing a clause clearly does not affect the embedding. The remaining transformations are of the following type:

(i) for given x, add $(xab) \wedge (x\neg ab) \wedge (x\neg bc) \wedge (x\neg b\neg c)$
(ii) for given x and y appearing in the same clause, add $(xy\neg a) \wedge (abc) \wedge (a\neg bc) \wedge (a\neg cd) \wedge (a\neg c\neg d)$
(iii) for given x, add $(xab) \wedge (xab)$.

Figure 8 shows how we can make these changes while maintaining proper nesting. In (i) vertices a, b and c are placed near x, so that x can still be linked to any other clause. In (ii) vertex a is placed near x, such that the clause $(xy\neg a)$ can take the place of the previous clause in which x and y appeared. We place b, c and d near a as we did in (i). In (iii) we place a and b near x. In this way, x can still be connected to any clause. □

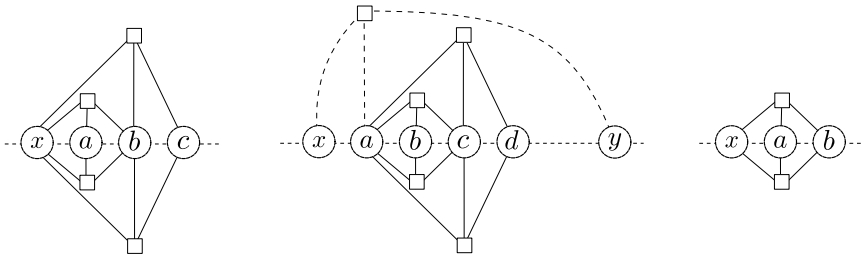

Fig. 8: Transformations corresponding to cases (i), (ii) and (iii).

Proof. (Lemma 5)

Both the variable and clause gadgets can be rotated, they can be made arbitrarily small and the bridges connecting them can be arbitrarily narrow. It remains to be shown that edges can emanate from a vertex at any required angle. In a planar circuit, we can move the clauses arbitrarily close to the line on which the variables lie, thereby making the three angles between the bridges of a clause $90° - \varepsilon_1$, $90° - \varepsilon_2$ and $180° + \varepsilon_3$, with ε_1, ε_2, ε_3 positive and arbitrarily close to zero, such that $\varepsilon_1 + \varepsilon_2 - \varepsilon_3 = 0$.

We then show by construction that a clause gadget can be stretched such as to have two angles of 85° and one angle 190° (Fig. 9). Since the gadget having these angles can be stretched continuously at any bridge to the symmetric gadget with all angles having 120°, a clause with angles $90° - \varepsilon_1$, $90° - \varepsilon_2$, $180° + \varepsilon_3$ can clearly be represented.

For variable gadgets we can prove something stronger: we can represent any angle in the interval $(0°, 360°)$ between neighboring bridges. To see this, consider a wire-piece between two neighboring bridges that does not bend at all, in which case the bridges are parallel. To produce a full circle, 16 vertices are sufficient, as seen in Fig. 3(b). Therefore, if we enforce that the number of vertices between two bridges is greater than 16, we can represent any angle. □

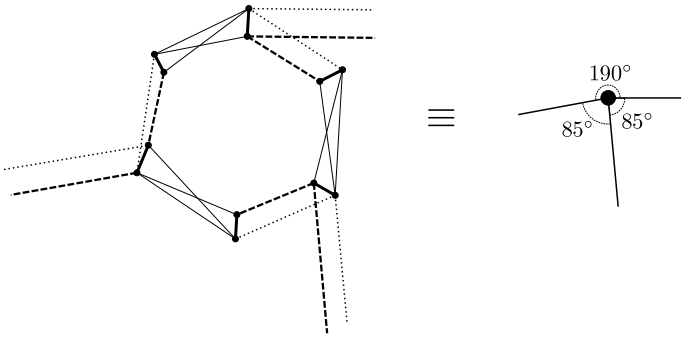

Fig. 9: A stretched clause.

Irrelevant edges. Let us count the number of vertices in the resulting MADT instance. Clause gadgets have 12 vertices each, as seen in Fig. 3(c), therefore the total number of vertices due to clauses is $12n_c$. The number of vertices in variable gadgets is $3n_c(N+2)$ (for all $3n_c$ bridges we have 2 base vertices and N vertices separating it from the next bridge in clockwise order). The number of all vertices in the construction is thus $18n_c + 3n_cN$ which is strictly smaller than $4n_cN$, assuming that $N > 18$.

Since the whole construction is connected and non-irrelevant edge weigths are at most 1, the longest possible distance is smaller than $4n_cN$. Thus, we can set the weights of irrelevant edges $\sigma = (4n_cN)^3 > \mathcal{W}(T)$. In this way the irrelevant edges never contribute to the cost, therefore we can simply ignore them. Note that the weights σ violate the triangle inequality.

Proof. (Lemma 6)

Suppose a variable is in impure state. Start with a dashed edge and follow the loop in clockwise order. At some point we switch to dotted edges. Where this happens, we have a local structure that we call *bubble*. There can be two different types of bubble, depending on whether the transition occurs at a bridge or somewhere else in the wire-piece. As we continue on the loop, we have to switch back to dashed edges somewhere. Where this happens, locally we have a *hole*. Figure 10 shows these local features. It is easy to see that if a variable gadget is impure, it has to have at least one hole and at least one bubble. What we show is that we can remove a hole and a bubble while decreasing the cost of the triangulation. When all holes and bubbles are removed, the triangulation is pure.

We consider a sequence consisting of a hole, a pure piece of wire and a bubble (of either type). The pure piece of wire might have several attached bridges along the way. We flip each flippable edge of the pure wire-piece (including any bridges along the way), thereby absorbing the hole and the bubble at the ends (Fig. 10(d)). Now we look at the change in cost due to this operation. We count

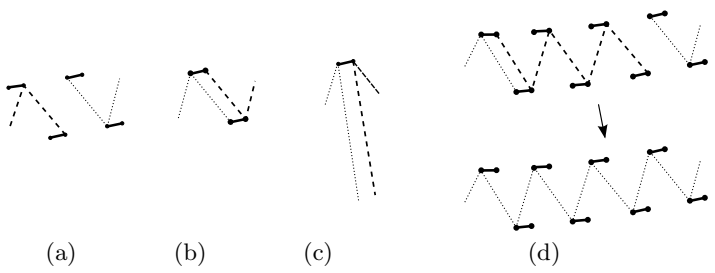

Fig. 10: (a) Hole. (b) Bubble. (c) Bubble at bridge. (d) Removing a bubble and a hole.

first the distances that might have become larger, then we count the distances which provably became smaller.

Take any two vertices and the shortest path between them. If a pure wire-piece contained in this path is flipped, the distance between the two endpoints can increase by at most 2 (we can simulate the old path with the new path and two extra edges of length 1 at both ends). If a bridge along the path is flipped (or partially deleted, if a bridge-bubble was removed), this can also increase any distance by at most 2 (we can simulate the old bridge using the new bridge and two extra edges at the bridge endpoints of length 1 each). If a bridge is flipped, this can force a change within the clause to which it is connected. In Fig. 3(c) we can verify that in any triangulation of a clause the distance between two vertices is less than 5. Therefore, a single clause can increase a path that somehow intersects it by less than 5 during this transformation. The total number of clauses is n_c and the number of bridges $3n_c$, therefore the penalty on any distance between two points is at most $2 + 3n_c \cdot 2 + n_c \cdot 5 = 11n_c + 2$. The number of distances is less than $\binom{4n_cN}{2}$, therefore the total penalty due to removing one bubble and one hole is smaller than $88{n_c}^3N^2 + 16{n_c}^2N^2 < 100{n_c}^3N^2$ (assuming $n_c > 1$).

Now we look at distances that provably decrease with the transformation. We can assume that between two neighboring bridges the variable gadget contains a single hole. If there were more, some vertices would be isolated and we would have to cross an irrelevant edge, incurring a cost of σ. In this case, removing the hole would obviously decrease the cost by making the construction connected using only non-irrelevant edges.

Consider now such a hole on the wire between two bridges (Fig. 11). Denote by n_1 the number of vertices between the previous bridge (in clockwise order) and the hole and by n_2 the number of vertices between the hole and the next bridge. Assume w.l.o.g. that $n_1 \leq n_2$. Depending on the state of the bridges, $n_1 + n_2$ can take the values N, $N + 1$ or $N + 2$. It follows that $n_2 \geq \frac{N}{2}$ and $n_1 \leq \frac{N}{2} + 1$. Among the n_2 vertices after the hole, consider the $\frac{N}{8}$ vertices closest to the hole and denote this set by A. On the other side of the other bridge denote the $\frac{N}{8}$ vertices closest to the bridge by B (Fig. 11(a)). Assume, for now, that B

is free of holes. We will treat the case when B has a hole, later. To make things simpler, we choose N to be divisible by 16. The following small lemma will help us with the computations. Its validity can be seen by a simple analysis of cases (see Fig. 4(a) and Fig. 10(d)).

Lemma 10. *If M is the longest distance between two vertices in a hole-free wire-piece with k vertices (possibly containing a bubble), then $\frac{k}{2} \leq M \leq \frac{k}{2} + 2$ holds.* □

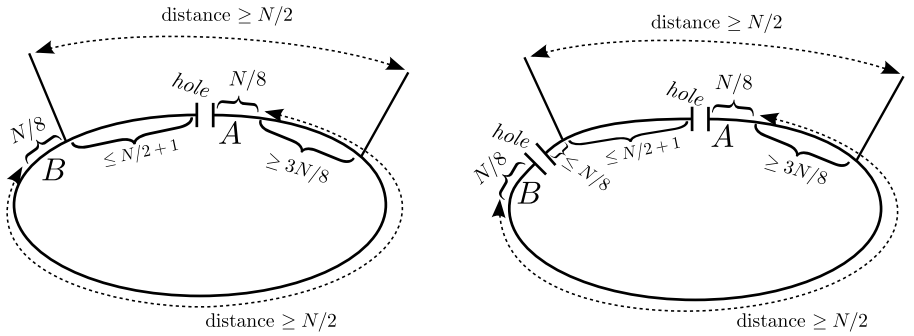

Fig. 11: Sketch of a variable gadget. A and B are brought closer by the removal of the hole(s).

We know that all variables have at least two bridges (due to R2) and all bridges of a variable are connected to different clauses (due to R1). Let us look at the minimum distance from a vertex in A to a vertex in B while the hole is still there. In any direction we have to go through a wire containing at least N vertices, therefore the distance is at least $\frac{N}{2}$ (due to Lemma 10). When the hole is removed, the distance between a vertex in A and a vertex in B is at most $\frac{3N}{8} + 3$.

If there was a hole within the $\frac{N}{8}$ vertices that we labeled as B, we take instead the $\frac{N}{8}$ vertices immediately before this hole and label them as B instead (Fig. 11(b)). In this case we can still claim that the distances between A and B are at least $\frac{N}{2}$ before the transformation. Now we remove both holes (and the two corresponding bubbles), potentially inflicting twice the penalty which we bounded from above by $100{n_c}^3N^2$. After removing both holes, distances between A and B are at most $\frac{7N}{16} + 3$. We get a decrease in cost of at least $\frac{N}{16} - 3$ for $\frac{N}{8} \cdot \frac{N}{8}$ pairs.

If we enforce $N > 5 \cdot 10^5 {n_c}^3$, we get a net decrease in cost due to the removal of the hole(s). Therefore, we can transform any impure triangulation into a pure one of lower cost. □

Proof. (Lemma 7)

The high level idea is the following: we first count the distances that can be smaller in T_{nonSAT} than in T_{SAT} and we bound their contribution to $\mathcal{W}$. Then we count those distances that are provably smaller in T_{SAT} than in T_{nonSAT} and we add up the differences. The crucial fact that makes the proof possible is the following: T_{nonSAT} has at least one clause with all three literals *false*. For this clause, crossing the gadget from one bridge to another has a cost of at least $1 + 2\varepsilon$. In T_{SAT} clause crossings have cost at most 4ε (Fig. 12). Using the fact that each clause crossing participates in $\Omega(N^2)$ distances, given bounds on N and ε we obtain the required bound on the difference between the costs.

Figure 12 shows the optimum triangulation of the clause gadgets in each possible assignment, ignoring symmetric cases. We are interested in the distances between the endpoints of the three bridges in the clause. These are summarized in the bottom row of Fig. 12. The triangulations are optimal in the sense that no other triangulation can achieve a lower distance between any of the bridge endpoints. These will be the triangulations used in T_{SAT}, but in T_{nonSAT} we will implicitly consider other triangulations of the clauses as well. Intuitively, it is clear that nonsatisfied clauses $\{F, F, F\}$ are costlier to cross than satisfied ones, and indeed, this is what makes our reduction possible. Now we make this intuition more precise.

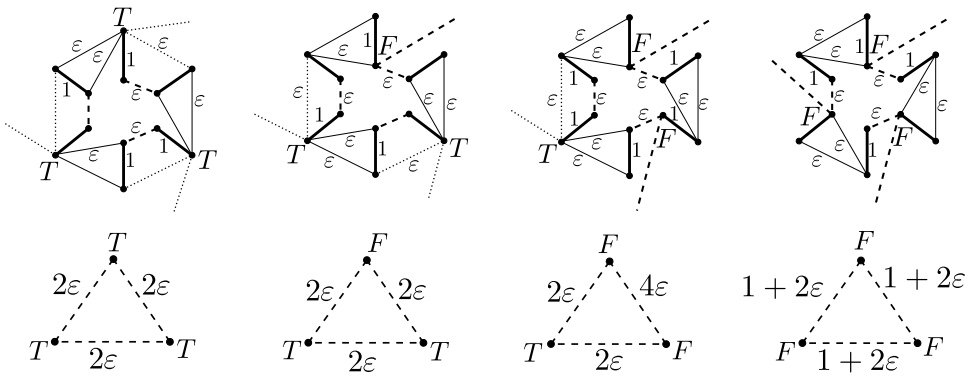

Fig. 12: (top) Optimal triangulation of a clause. (bottom) Cost of crossing a clause.

Every distance between two vertices is of one of the following types: (i) between vertices of the same clause, (ii) between vertices of the same variable, (iii) between a vertex from a clause and a vertex from a variable, (iv) between vertices of different clauses, and (v) between vertices of different variables.

We go through the five types of distances and denote their contribution to the cost by $\mathcal{W}_1, \ldots, \mathcal{W}_5$. In each of the five cases we want to compare the cost of T_{SAT} with the cost of T_{nonSAT}.

(i) Within a single clause, even in the most unfavorable triangulation, the distance between any two vertices is less than 5, therefore $\mathcal{W}_1 < n_c\binom{12}{2}5 = 330n_c$.

(ii) Variable gadgets in the two different states are isomorphic, therefore $\mathcal{W}_2(T_{\text{SAT}}) = \mathcal{W}_2(T_{\text{nonSAT}})$.

(iii) There are less than $(4n_cN)(12n_c)$ such distances. If we look at one shortest path as we move from a satisfying to a non-satisfying assignment, the path length can decrease by at most 1 at both endpoints. Variable-crossings and bridge-crossings along the way maintain their length and clause-crossings can decrease by at most 2ε each (compare crossing costs of clauses in Fig. 12). Therefore $\mathcal{W}_3(T_{\text{SAT}}) - \mathcal{W}_3(T_{\text{nonSAT}}) < (2 + 6n_c\varepsilon)(4n_cN)(12n_c)$, which, assuming $\varepsilon < \frac{1}{6n_c}$, is less than $144{n_c}^2N$.

(iv) By a similar argument, $\mathcal{W}_4(T_{\text{SAT}}) - \mathcal{W}_4(T_{\text{nonSAT}}) < (2 + 6n_c\varepsilon)(12n_c)(12n_c) < 432{n_c}^2$, assuming $\varepsilon < \frac{1}{6n_c}$.

(v) This part is the crucial one, since it contributes the highest order term in N to the cost. Our goal is to show that $\mathcal{W}_5(T_{\text{nonSAT}}) - \mathcal{W}_5(T_{\text{SAT}}) = \Theta(N^2)$, outweighing the other four differences which are all $\mathcal{O}(N)$.

Let us look at the distance between two vertices, p_x and p_y from different variable gadgets (Fig. 13). Let $d(p_x, p_y)$ be their distance in T_{SAT} and $d'(p_x, p_y)$ their distance in T_{nonSAT}.

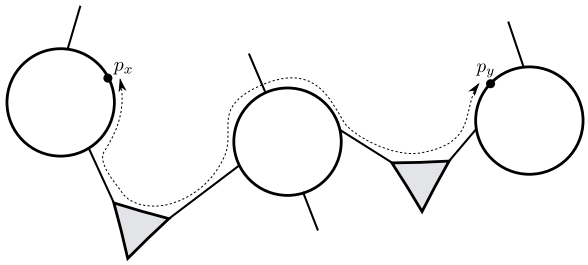

Fig. 13: Shortest path between p_x and p_y. Variables appear as circles, clauses as triangles.

We denote variable gadgets as $V_1, \ldots, V_{n_v}$ and we write the cost due to distances between vertices from different variables as:

$$\mathcal{W}_5(T_{\text{SAT}}) = \sum_{\substack{p_x, p_y: \\ p_x \in V_i,\ p_y \in V_j \\ 1 \le i < j \le n_v}} d(p_x, p_y).$$

In Fig. 14 we see that every vertex has a natural *neighbor*, the vertex to which it is connected by a thick solid edge. We denote the neighbors of p_x and p_y as $\overline{p_x}$ and $\overline{p_y}$, respectively. We define the distance between two pairs of neighboring points as follows:

$$d\big([p_x\overline{p_x}], [p_y\overline{p_y}]\big) = d(p_x, p_y) + d(\overline{p_x}, p_y) + d(p_x, \overline{p_y}) + d(\overline{p_x}, \overline{p_y}).$$

One vertex out every pair of neighbors is a *leaf vertex*, in the sense that a path from that vertex to any other vertex goes through its neighbor. In Fig. 14,

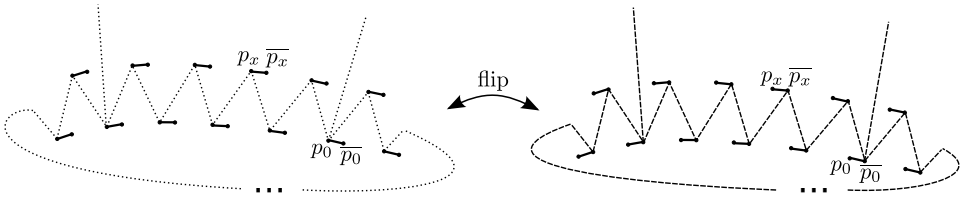

Fig. 14: Vertices in a bridge-to-bridge portion of a variable gadget.

left, $\overline{p_x}$ and $\overline{p_0}$ are leaf vertices, but when the variable is flipped into the other pure state (Fig. 14, right), the situation reverses and p_x and p_0 become leaves. We can simplify the distance between pairs of points as follows:

$$d\big([p_x\overline{p_x}], [p_y\overline{p_y}]\big) = 4\ \phi(p_x, p_y) + 4,$$

where ϕ is the distance between the non-leaf members of both pairs, or more precisely:

$$\phi(p_x, p_y) = \min\big(d(p_x, p_y),\ d(\overline{p_x}, p_y),\ d(p_x, \overline{p_y}),\ d(\overline{p_x}, \overline{p_y})\big).$$

Now we can write the relevant part of the cost in terms of ϕ:

$$\mathcal{W}_5(T) = \sum_{\substack{p_x, p_y: \\ p_x \in v_i,\ p_y \in v_j \\ 1 \le i < j \le n_v}} \big(\phi(p_x, p_y) + 1\big).$$

Let $\phi(p_x, p_y)$ denote the distance defined above in T_{SAT} and $\phi'(p_x, p_y)$ the corresponding distance in T_{nonSAT}. We want to bound $\phi - \phi'$ from above. Let us decompose $\phi(p_x, p_y)$ into components. Remember that ϕ is the distance between two non-leaf points, i.e., the length of the shortest path between them. Such a path goes from p_x to a bridge, then crosses a number of bridges, clauses and variables, arrives to the target variable, and goes from the bridge to p_y. Observe that the first and last components (endpoint to bridge) do not change with the flipping of a variable. This can be seen in Fig. 14: the distance $d(p_x, p_0)$ on the left and the distance $d(\overline{p_x}, \overline{p_0})$ on the right are equal. Variable-crossing costs do not change either, a variable bridge-to-bridge portion always has distance $\frac{N}{2} + 1$ and neither do bridge-crossings which always cost ε.

The only difference in cost between ϕ and ϕ' is due to clause crossings. Whereas T_{SAT} contains only clauses of the type $\{T, T, T\}$, $\{T, T, F\}$, $\{T, F, F\}$, in T_{nonSAT} we have at least one $\{F, F, F\}$ clause. Thus, according to Fig. 12, the maximum cost of a crossing in T_{SAT} is 4ε and the minimum cost of a crossing in T_{nonSAT} is 2ε. A shortest path can cross each clause only once, otherwise there would exist a shortcut. Since there are n_c clauses in total, we obtain the bound:

$$\phi(p_x, p_y) \le \phi'(p_x, p_y) + 2n_c\varepsilon.$$

In $\mathcal{W}_5$ we have at most $\binom{3n_c}{2}(N+2)^2$ distances, each of which can be shorter by at most $2n_c\varepsilon$ in T_{nonSAT} than in T_{SAT} (provided that we group distances

four-by-four as explained above and we average over the groups). The number of distances (assuming $N \geq 12{n_c}^2$) is less than $4{n_c}^2N^2$.

Now let us look at distances that are provably larger in T_{nonSAT} than in T_{SAT}. We know that there is at least one clause crossing that has cost $1+2\epsilon$ in T_{nonSAT} and cost at most 4ε in T_{SAT}. This clause crossing is thus at least $1-2\varepsilon$ costlier in T_{nonSAT} than in T_{SAT}. For every clause crossing there are at least $\frac{N}{4} \cdot \frac{N}{4}$ shortest paths going through that crossing, regardless of the states of the variables, i.e., both in T_{nonSAT} and in T_{SAT}. This fact is illustrated in Fig. 15 (sets A and B). In this way we get that there are at least $(\frac{N}{4})^2$ distances that contribute at least $1-2\varepsilon$ more to $\mathcal{W}_5(T_{\text{nonSAT}})$ than to $\mathcal{W}_5(T_{\text{SAT}})$.

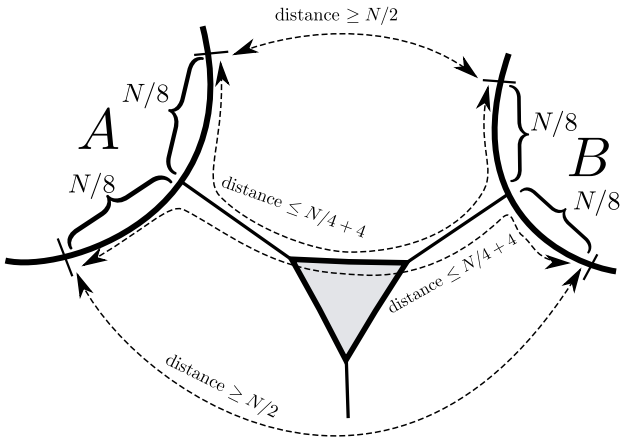

Fig. 15: Shortest paths that have to go across a given clause crossing.

Now we have all the ingredients to compare $\mathcal{W}(T_{\text{nonSAT}})$ and $\mathcal{W}(T_{\text{SAT}})$:

$$\begin{aligned}
\mathcal{W}(T_{\text{nonSAT}}) - \mathcal{W}(T_{\text{SAT}}) = \; & \mathcal{W}_1(T_{\text{nonSAT}}) - \mathcal{W}_1(T_{\text{SAT}}) + \mathcal{W}_2(T_{\text{nonSAT}}) - \mathcal{W}_2(T_{\text{SAT}}) \\
& + \mathcal{W}_3(T_{\text{nonSAT}}) - \mathcal{W}_3(T_{\text{SAT}}) + \mathcal{W}_4(T_{\text{nonSAT}}) - \mathcal{W}_4(T_{\text{SAT}}) \\
& + \mathcal{W}_5(T_{\text{nonSAT}}) - \mathcal{W}_5(T_{\text{SAT}}) \\
\geq \; & -330n_c - 144{n_c}^2N - 432{n_c}^2 - 8{n_c}^3N^2\varepsilon + (\frac{N}{4})^2(1-2\varepsilon) \\
\geq \; & \frac{N^2}{32}. \\
& \qquad \text{(assuming } N > 5 \cdot 10^5{n_c}^3 \text{ and } \varepsilon < \tfrac{1}{N^2}\text{)}
\end{aligned}$$

□

Proof. (Lemma 8)

We go through the same computations as for Lemma 7 and use the fact that between two satisfying assignments a clause crossing can change cost by at most 2ε. □

Baseline Triangulation. Our goal is to generate $\mathcal{W}^\star$ somewhere in the gap between the costs of satisfying and non-satisfying triangulations. It would be sufficient to generate a triangulation corresponding to a satisfying assignment and add $150{n_c}^2N$ to its cost. We do not even know, however, whether a satisfying assignment exists.

Instead, we construct a simpler triangulation that we call *baseline*: we assign to each variable an arbitrary truth value and triangulate the variable gadgets and attached bridges accordingly. Then we replace each clause with the baseline gadget of Fig. 16, connecting the three vertices of the triangle to the bridge endpoints that that were supposed to connect to that clause. We note that many other configurations would work similarly well as a baseline gadget. For the described construction we can compute $\mathcal{W}(T_{\text{baseline}})$ using an all-pairs shortest path algorithm.

Proof. (Lemma 9)

The computations are similar to those in the previous proofs (we look at the five different types of distances):

(i) Within a single clause of T_{baseline} the distance between any two vertices is less than 2, therefore $\mathcal{W}_1(T_{\text{baseline}}) < n_c\binom{12}{2}2 = 132n_c$.
(ii) Here also $\mathcal{W}_2(T_{\text{baseline}}) = \mathcal{W}_2(T_{\text{SAT}})$.
(iii) Here also $\left|\mathcal{W}_3(T_{\text{baseline}}) - \mathcal{W}_3(T_{\text{SAT}})\right| < (2+6c\varepsilon)(4n_cN)(12n_c) < 144{n_c}^2N$.
(iv) Here also $\left|\mathcal{W}_4(T_{\text{baseline}}) - \mathcal{W}_4(T_{\text{SAT}})\right| < (12n_c)(12n_c)(2+6n_c\varepsilon) < 432{n_c}^2$.
(v) Here also, clause crossings can change by at most 2ε, therefore $\left|\mathcal{W}_5(T_{\text{baseline}}) - \mathcal{W}_5(T_{\text{SAT}})\right| < 8{n_c}^3N^2\varepsilon$.

Overall we find that $\left|\mathcal{W}(T_{\text{baseline}}) - \mathcal{W}(T_{\text{SAT}})\right| \leq 150{n_c}^2N$. □

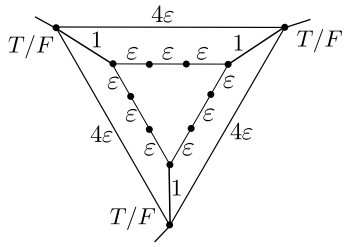

Fig. 16: Clause gadget in baseline triangulation.